Section 508 Web Accessibility Standard

The 508 accessibility standards come from the federal Rehabilitation Act Amendments of 1998, which requires the U.S. federal government to make sure that any electronic or information technology they purchase will allow disabled citizens and employees to access or use the technology. Many states, including Missouri, have adopted these standards into state law.

§ 1194.22 Web-based intranet and internet information and applications.

(a) A text equivalent for every non-text element shall be provided (e.g., via "alt", "longdesc", or in element content).

(b) Equivalent alternatives for any multimedia presentation shall be synchronized with the presentation.

(c) Web pages shall be designed so that all information conveyed with color is also available without color, for example from context or markup.

(d) Documents shall be organized so they are readable without requiring an associated style sheet.

(e) Redundant text links shall be provided for each active region of a server-side image map.

(f) Client-side image maps shall be provided instead of server-side image maps except where the regions cannot be defined with an available geometric shape.

(g) Row and column headers shall be identified for data tables.

(h) Markup shall be used to associate data cells and header cells for data tables that have two or more logical levels of row or column headers.

(i) Frames shall be titled with text that facilitates frame identification and navigation.

(j) Pages shall be designed to avoid causing the screen to flicker with a frequency greater than 2 Hz and lower than 55 Hz.

(k) A text-only page, with equivalent information or functionality, shall be provided to make a web site comply with the provisions of this part, when compliance cannot be accomplished in any other way. The content of the text-only page shall be updated whenever the primary page changes.

(l) When pages utilize scripting languages to display content, or to create interface elements, the information provided by the script shall be identified with functional text that can be read by assistive technology.

(m) When a web page requires that an applet, plug-in or other application be present on the client system to interpret page content, the page must provide a link to a plug-in or applet that complies with §1194.21(a) through (l).

(n) When electronic forms are designed to be completed on-line, the form shall allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.

(o) A method shall be provided that permits users to skip repetitive navigation links.

(p) When a timed response is required, the user shall be alerted and given sufficient time to indicate more time is required.

Resources:

http://www.section508.gov

Digital Media Developers Web Guidelines – Accessibility: http://dmd.mo.gov/guidelines/accessibility.pdf

Missouri IT Accessibility Standard
http://www.oa.mo.gov/itsd/cio/standards/ITGS0003.pdf

Standards-Based Coding Best Practices

The need to make Missouri Web sites accessible drives the need to follow standards in web design. A well-designed standards-compliant site will be mostly accessible because it will be a basic, well-formed text page with some presentation styling.

XHTML

There are ten major requirements for XHTML that are not in HTML.

1. You must use a proper DTD.
2. You must use a proper namespace.
3. You must declare your content type.
4. You must make all your tags lowercase.
5. You must quote all attribute values.
6. All attributes require values.
7. You must close all tags.
8. Even empty tags, like <br>.
9. You can't use double dashes in comments, and
10. You need to encode all < and & characters.

Document Type Definition (DTD)

Each XHTML page should have a Document Type Definition (DTD or Doctype) as the first line of the page.

The DTD tells the browser what set of tags to interpret in your page. XHTML specifies three DTDs, so authors must include one of the following document type declarations in their documents. The DTDs vary in the elements they support.

The XHTML Strict DTD includes all elements and attributes that have not been deprecated or do not appear in frameset documents. For documents that use this DTD, use this document type declaration:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

The XHTML Transitional DTD includes everything in the strict DTD plus deprecated elements and attributes (most of which concern visual presentation). For documents that use this DTD, use this document type declaration:

```
<!DOCTYPE HTML "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional. dtd">
```

The XHTML Frameset DTD includes everything in the transitional DTD plus frames as well. For documents that use this DTD, use this document type declaration:

```
<!DOCTYPE HTML "-//W3C//DTD XHTML 1.0 Frameset//EN" "http: //www . w3 .
org/TR/xhtml1/DTD/xhtml1-frameset . dtd">
```

If you are using HTML, DTDs can be found at: http://www.w3.org/TR/html401/sgml/dtd.html

The DTDs tell the browser how to interpret the code in your page. XHTML-Transitional is closest to regular HTML, and is most forgiving of invalid code.

Most web editors will either insert a DTD automatically or can be configured so the main template will automatically insert one.

Strict or Transitional?

Transitional (also known as “quirks”) mode kicks in whenever a DTD is not declared, or if your document does not validate. It is the default DTD for Internet Explorer 6. Writing to a transitional standard allows you to use some deprecated attributes, like “ ”

Strict mode doesn't cut you any slack. Your code must validate or your page will get displayed using quirks mode. This means you have to be a little more careful when creating your pages, but you will have greater control over your page. IE6 will display strict mode properly, so you will have a consistent look no matter which browser your users use.

So which one do you use? If you are uncomfortable with changing the way you have always created pages, Transitional mode may be the one for you. It will give you a little more time to clean up sloppy code and tighten up your pages. But remember, it is
transitional. You really need to think about switching over to Strict sooner or later.

If you make the jump into Strict mode, you may be confused early on as you get used to the no-nonsense rules, but they soon become second nature. Some development tools, like Dreamweaver MX, will automatically convert your pages to comply with your DTD.

Namespace

The namespace is an extension of the basic <html> tag found at the top of each old web page. It follows immediately after the DTD statement, and looks like this:

```
      <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
      lang="en">
```

The first part of this statement, *xmlns="http://www.w3.org/1999/xhtml*", tells the browser
where to find the namespace on the web.
The other two parts, *xml:lang="en" lang="en"*, indicate that the XML and page language is English.

Content Type

The actual W3C XHTML specification calls for an <xml> statement above the DTD, where you declare the character encoding for the document. Unfortunately, if you do this, IE will automatically render the page in quirks mode, which can mess up your layout. Other browsers may actually crash.

Instead, you will insert a meta tag in the head of the document, like this:

```
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
```

In this example, the document uses the ISO-8859- 1 character set (also known as Latin-1). There are other character sets available if you are coding in other languages or need an internationalized character set.

Lowercase Tags

XHTML is case-sensitive. All elements and attributes (tags and properties) must be in lowercase in order to validate. For example, this is invalid code:

```
      <OL CLASS="bigfont"><LI OnMouseOver="doJS">Item 1</LI><LI>Item
      2</LI></OL>
```

but this is valid code:

```
      <ol class="bigfont"><li onmouseover="doJS">Item 1</li><li>Item
      2</li></ol>
```

Quote Attribute Values

This is pretty simple. Instead of having code like this:

```
      <table cellspacing=0 cellpadding=2 border=1>
```

You will need to use quotes:
```
<table cellspacing="0" cellpadding="2" border="1">
```

You also need to separate each attribute with a space.

All attributes require values

Any attribute that was just a word in HTML will need to have a value declared as well. HTML:
```
<input type=”radio” name=”radio1” value=”5” selected>
```

XHTML:
```
<input type=”radio” name=”radio1” value=”5”
selected=”selected”>
```

Close all tags

HTML let you assume a tag closed when you started a new one, like this:
```
<p>The quick red fox jumped over the lazy brown dog. <p>She sells
seashells by the seashore.
```

XHTML makes you close those tags, which is a good habit to get into anyway:
```
<p>The quick red fox jumped over the lazy brown dog.</p> <p>She sells
seashells by the seashore.</p>
```

Close the empty tags, too

Tags like `<br>, <hr>` and `<img>` do not have a corresponding closing tag (like `</br>, </hr>` or `</img>).`

In XHTML, you close the tag by adding a space and a forward slash before the closing >, like this: `<br />, <hr />` and `<img />`

No double dashes in comments

XHTML only allows a double dash (“--”) at the beginning and the end of a comment. So this is not allowed:

```
<!--This is a bad comment—no fooling. -->
```

But you can put a space between the dashes, so this is valid: `<!--This is a valid comment- -for real. -->`

Encode special characters

If you use an ampersand, greater than sign, less than sign or quote (“&”, “>”, “<” and “””, respectively) in your text, you will need to encode them like this: &amp;, &gt;, &lt; and &quot;.

Those characters are reserved in XML, but XHTML validators will let you off with a warning

Cascading Style Sheets (CSS)

- Use linked style sheets rather than embedded styles - referencing an external file will you give the maintenance benefits of being able to update the look of your entire site with a single update in addition to an improvement in load time. Individual page authors can create additional embedded styles for their own pages when necessary to override the centralized (linked or @import) style.

- Pages must continue to work when style sheets are disabled. Retaining a decent presentation without the style sheet is mandatory to support disabled users and older browsers.

- If possible, have a single style sheet file for the whole site; or at most, separate files for a small number of sub-site categories. Having one set of styles used across the site will give the whole site a consistent look and formatting, and make the site easier to use.

- Do not use absolute font sizes - specify all text relative to the base font size defined by the user's preference setting (i.e., use 1.0 em or 100% instead of 12px or 12pt. This will allow automatic font resizing based on browser options selected.)

- Multiple style sheets - make sure to use the same class names for the same concept in all of the style sheets. Content creators using two or more style sheets will be confused if different classes are used for the same thing or if one style sheet has a class that is missing in the other style sheet even though the concept applies in both cases. If you have a class for the name of the author of a document, then all of your style sheets should have this class, even though it may be defined to render differently, as appropriate for the different kinds of documents.

- Print stylesheets - with a standards-based site the printer-friendly template is replaced by a print stylesheet, which is applied to the regular page and only used when that page is actually printed. Those linking directly can be forwarded to the regular article page. Not only can bandwidth be saved by not duplicating files, but site managers can also ensure that site branding is maintained and that all visitors are able to browse and search the site from all pages.

Summary

Keep the following in mind when creating your pages:

1. Use valid HTML.
2. Content should be separate from style.
3. Don't make pages too long so that there is endless scrolling, unless you provide "anchors" and links to aid the reader in navigation.
4. Avoid using deprecated tags.
   - The following tags have been deprecated and should no longer be used. HTML 4.01 will allow you to use them, but XHTML 1.0 forbids them:
     - CENTER
     - FONT
     - U (underline)
     - APPLET

     These tags should be replaced with appropriate style sheets or standards compliant tags.

Resources:

Digital Media Developers Web Guidelines – Standards: http://dmd.mo.gov/guidelines/introStds.pdf#page=3

State of Missouri Architecture Committee Interface Domain:
http://www.oa.mo.gov/itsd/cio/architecture/domains/interface/CC-CSS091905.pdf